Claims

[c1] 1.A compiling method for converting into object code a program written in source code comprising the steps of:
allocating registers for a program to be compiled; and
generating object code based on the register allocation,
wherein said step of allocating registers includes the steps of
allocating logical registers for instructions in said program, and
performing mapping between said logical registers and physical registers, so that said physical registers that are live at a procedure call in said program to be compiled are allocated from the bottom of the register stack.

[c2] 2.The compiling method according to claim 1, wherein, at said mapping step, allocation is done so that logical registers that are live across more procedure calls are first allocated.

[c3] 3.The compiling method according to claim 1, wherein, at said mapping step, allocation is done, so that logical registers that are live across a procedure call at which fewer logical registers are live at the same time are first allocated.

[c4] 4.A code generation method for generating code for a program that controls a computer comprising the steps of:
generating code while confirming that registers are allocated for a predetermined instruction; and
upon the calling of the procedure, so long as there is a vacancy in operation resources, copying said registers residing in the register stack, to free registers located at the bottom of said register stack.

[c5] 5.A method, for employing a stack register when a processor with a register stack executes a program, comprising the steps of:
when a different procedure is called in a predetermined procedure, reallocating registers that are allocated for the execution of said predetermined procedure and are live when said different procedure is called, and calling said different procedure; and
upon the return from said different procedure, restoring the register image to the state immediately before the reallocation.

[c6] 6.The stack register employment method according to claim 5, wherein said step of reallocating said registers and calling said different procedure includes the steps of:
sorting and reallocating, from the bottom of said register stack, said registers that are live when said different procedure is called.

[c7] 7.A method, for employing a stack register when a program is executed by a processor with a register stack, comprising the steps of:
each time a procedure is called, packing and allocating existing logical registers;
performing said procedure, and restoring the register image to the state before the packing.

[c8] 8.A compiler for converting into machine language code the source code of a program written in a programming language, the complier comprising:
a register allocator, for allocating registers for instructions in said program to be compiled; and
a code generator, for generating object code based on the register allocation process performed by said register allocator,
wherein said register allocator allocates logical registers for instructions in said program to be compiled, and allocates, to physical registers, said logical registers that are allocated to said instructions of said program, so that said physical registers that are live at a procedure call in said program to be compiled are allocated from the bottom of the register stack.

[c9] 9.The compiler according to claim 8, wherein said register allocator allocates said logical registers and said physical registers first for an important portion of said program to be compiled; and wherein, while for a less important portion of said program, said code generator generates compensation code for allocation of said logical registers and for allocation of said physical registers for the important portion.

[c10] 10.A compiler for converting into machine language code the source code of a program written in a programming language, the complier comprising:
a register allocator, for allocating registers for instructions in said program to

be compiled; and

a code generator, for generating object code based on the register allocation process performed by said register allocator,

wherein said code generator generates code while confirming that registers are allocated for predetermined instructions, and

wherein, upon a procedure being called, said code generator, so long as there is a vacancy in operation resources, copies said registers residing in a register stack, to free registers that are located at the bottom of said register stack.

[c11] 11.A computer comprising:

input means, for entering source code of a program; and

a compiler, for compiling said source code and converting the compiled code into machine language code,

wherein, before a different procedure is called in a predetermined procedure of a program to be compiled, said compiler generates code for reallocating registers that are allocated for the execution of said predetermined procedure and that are live when said different procedure is called, and generates code, restoring the register image, upon the return from said different procedure, to the state immediately before the reallocation.

[c12] 12.A conversion program, for controlling a computer for conversion of a program to be executed, which permits said computer to perform, the conversion program comprising:

a process for allocating logical registers for instructions in said program to be executed;

a process for performing mapping between said logical registers and physical registers, so that said physical registers that are live at a procedure call in said program to be compiled are allocated from the bottom of the register stack; and

a process for generating object code based on the mapping process.

[c13]

13.A conversion program, for controlling a computer for conversion of a program to be executed, which permits said computer to perform, the conversion program comprising:

a process for generating code while confirming that registers are allocated for a

predetermined instruction; and
a process for, upon the calling of the procedure, so long as there is a vacancy in operation resources, copying said registers residing in the register stack, to free registers located at the bottom of said register stack.

[c14] 14.A program, for controlling a computer to execute an operation, which permits said computer to perform, the conversion program comprising:
a process for, when a different procedure is called in a predetermined procedure, reallocating registers that are allocated for the execution of said predetermined procedure and that are live when said different procedure is called, and calling said different procedure; and
a process for, upon the return from said different procedure, restoring the register image to the state immediately before the reallocation.

[c15] 15.A storage medium on which a conversion program is stored that controls a computer for conversion of a program to be executed, said conversion program permitting said computer to perform, the conversion program comprising:
a process for allocating logical registers for instructions in said program to be executed;
a process for performing mapping between said logical registers and physical registers, so that said physical registers that are live at a procedure call in said program to be compiled are allocated from the bottom of the register stack; and
a process for generating object code based on the mapping process.

[c16] 16.A storage medium on which a conversion program is stored that controls a computer for conversion of a program to be executed, said conversion program permitting said computer to perform, the conversion program comprising:
a process for generating code while confirming that registers are allocated for a predetermined instruction; and
a process for, upon the calling of the procedure, so long as there is a vacancy in operation resources, copying said registers residing in the register stack, to free registers located at the bottom of said register stack.

[c17] 17.A storage medium on which a program is stored that controls a computer to execute an operation, said program permitting said computer to perform, the

program comprising:

a process for, when a different procedure is called in a predetermined procedure, reallocating registers that are allocated for the execution of said predetermined procedure and that are live when said different procedure is called, and calling said different procedure; and

a process for, upon the return from said different procedure, restoring the register image to the state immediately before the reallocation.